ST Xpay User Guide

Version 3.51

securetrading

Copyright

© SecureTrading® 2010. All rights reserved. No part of this document may be photocopied, reproduced, stored in a retrieval system or transmitted in any form or by any means whether electronic, mechanical or otherwise without the prior written permission of SecureTrading Ltd.

Disclaimer

This document is for informational purposes only. SecureTrading make no warranties, express or implied, through the distribution of this document. No warranty of accuracy is given concerning the contents of the information contained in this publication. To the extent permitted by law no liability (including liability to any person by reason of negligence) will be accepted by SecureTrading, its subsidiaries or employees for any direct or indirect loss or damage caused by omissions from or inaccuracies in this document.

SecureTrading reserves the right to revise the content without obligation to notify any person of such changes.

SecureTrading is the registered trademark of SecureTrading Group Ltd.

Document revised on 03-Dec-2010.

Encryption

Cryptix General License

Copyright (c) 1995 - 2004 The Cryptix Foundation Limited. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE CRYPTIX FOUNDATION LIMITED AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE CRYPTIX FOUNDATION LIMITED OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

1 Introduction

Thank you for choosing the SecureTrading Xpay client. This guide provides an introduction to the set-up and configuration of Xpay. This document must be read in conjunction with the ST XML Specification, available on the SecureTrading web site. For document locations, please refer to the Additional information section of this document.

Please note the example commands are shown for MS-DOS users. Other operating systems such as Unix will need to replace shell commands with the appropriate equivalent.

This guide is for users of the ST Xpay client only. If you are using ST Xpay4 please refer to the separate user guide.

2 Xpay Setup

2.1 Prerequisites:

- A compatible java runtime environment. We recommend using java 1.4.2. Other versions of java may be used at your discretion but no guarantee can be made of their compatibility.
- Access to a command prompt or shell
- Your SecureTrading Site Reference and Xpay certificate
- internet access from your server

You should have Java specified in the environment variable PATH. If Java has been installed to the default location, from the command prompt type:

For example:

```
c:
set path=%path%;"C:\j2sdk1.4.2_06\bin"
```

Note: Your Site Reference is as quoted in your SecureTrading New Merchant Account email. This is usually made up of an alphanumeric string corresponding to your business. Both your Certificate File and your Site Reference will be unique to your company and are *case sensitive*.

The running example in this document assumes your company has the Site Reference:
```
company1234
```
And you have been sent the certificate file (`.pem` file):
```
company1234xpaycerts.pem
```

If you don't know your Site Reference, please check first with the person who is named on the Merchant Agreement with SecureTrading. If your Site Reference is still unavailable, contact support@securetrading.com

If you do need to contact SecureTrading, then the site owner only should disclose the following information to help us find out your Site Reference:

- The company name of the Merchant
- The name of the person who signed the original Merchant Agreement

If you have not been issued with the certificate file, then please contact SecureTrading stating:
- Your Site Reference
- Which version of Java Runtime you are using
- Which Operating System you are using
- That you want to use Xpay and that you have your own Secure Server.

2.2 Installation

The examples are for MS-DOS users, but a similar approach is taken for users of other Operating Systems:

Unzip the entire contents of `xpay.zip` into a new directory, `c:\xpay`, on your server.

There are several ways to do this; the simplest way is using WinZip. If you do not have a copy of WinZip, or are unsure how to use it, please visit www.winzip.com for information and downloads.

The Xpay API requires the file `XPay.jar` to be in the CLASSPATH environment variable. To do this, you must set the CLASSPATH.

> E.g.   `set CLASSPATH=.;C:\xpay\XPay.jar`

Note: the above command is *__case sensitive__*.

2.3 Setting Permissions

Xpay will automatically check for updated versions when starting and therefore requires access to the internet on port 80 (the standard http port). Any file which is downloaded is signed by SecureTrading to ensure authenticity.
__Please Note:__ You should ensure that Xpay is run under a security manager to authenticate these signature files by following these steps:

The SecureTrading Xpay Public Certificate is in the file `securetradingxpay.cer` and needs to be imported as a trusted certificate. First, change to any directory on your server, which is not available to public users (`c:\keys` will be used for this example), and then create a new keystore using the standard java keytool program:
__Please Note:__ You must move `securetradingxpay.cer` into `c:\keys` before running the following commands.
E.g.

```
cd \keys
keytool –import –alias xpay –file securetradingxpay.cer –keystore mystore
```

You should ensure the keytool command is set in the PATH environment variable as above.

You will be prompted to enter a password for your keystore. If the store does not already exist you should choose your own password. Choose a good password to ensure only trusted users have access to the keystore.

You will then be asked if you wish to trust the Xpay certificate. You must be sure that any certificate you import is one you trust. Please check the fingerprint of the certificate to ensure it has not been tampered with:

```
Certificate fingerprints:
     MD5:  74:5F:14:9B:5B:13:E7:92:38:9F:49:8C:33:F3:A7:45
     SHA1: A5:2E:5D:A3:58:53:32:29:27:3D:7B:33:7C:84:A2:95:9D:B2:91:D4
```

If the output is not *__exactly__* as above you should *__not__* trust the certificate. Contact SecureTrading to get a new copy of the public certificate. If possible, state where you obtained the invalid certificate file.

Note, if you see the following fingerprint you are using an outdated expired certificate. Please contact support@securetrading.com to obtain the latest certificate.

```
MD5:   CA:B5:7F:80:92:47:77:44:97:D7:C2:CC:67:2F:90:CB
SHA1: AD:CB:7A:29:5A:25:AC:8C:B6:20:D5:5A:69:B6:89:9F:E7:74:AF:FD
```

Once you have validated the public certificate, type "`yes`" to confirm you wish to import the certificate as a trusted source.

2.4 Creating a Policy file

A policy is now needed to allow Xpay to run only if it is signed by SecureTrading. An example policy file is included with the Xpay distribution. If you created your keystore in a different directory or a different name to the example, you should modify the examplepolicy. Note that the location of the keystore is a URL and should therefore begin with "`file:/`" and directory separators should be forward slashes ("`/`") not backslashes ("`\`").

E.g. `file:/c:/keys/mystore`

Alternatively, you may create a new policy file with a text editor or by using the built-in java tool "`policytool`". You must be using a graphical operating system to use the policy tool.

2.5 **Running the Xpay client**

From the command prompt, change to your Xpay directory.
```
E.g.   c:
            cd \xpay
```

You must run Xpay using a security manager and the policy created above. This can be done dynamically by specifying the policy on the command line:

E.g.
```
java –Djava.security.manager –Djava.security.policy=c:\xpay\examplepolicy XPay
```

Note: the above command is *case sensitive*.

This command starts the client that establishes secure connections to SecureTrading's gateways; this client must be running permanently for the Xpay operation.

<u>Please Note:-</u> If you see a pop-up window select un-block.

The output should be similar to the following:

```
Checking for updates to XPay. Please wait...
Download complete
Client starting....
```

If the Xpay client fails to start, please refer to the Troubleshooting guide. See section 5 Troubleshooting.

<u>Important Note</u>: you should only have *exactly one* client running on your server.

2.5.1 Configuration file

ST Xpay supports an optional configuration file called "xpay.ini". Any configurable details can be stored in this file. When ST Xpay starts it searches in the current directory for the configuration file, if it finds one then the parameters in the file will be used, otherwise the defaults will be used.

Important

- To start ST Xpay with a configuration file you must be in the same directory as the configuration file when running the start-up script.
- The configuration file cannot be renamed

To use the configuration file the java security policy file must be configured to grant access to read the file. This is to enable ST Xpay read access to the file. Typically the following line is included in the java policy file:

- `permission java.io.FilePermission "xpay.ini", "read";`

In the xpay.ini file there should be one line for each configuration option. Each configuration option must be of the format `"name=value"`. If the xpay.ini file is missing, inaccessible or the value of an argument is invalid or missing, the default value will be used.

Currently the following options can be passed to ST Xpay:

Name	Value	Default	Description	Example
port	INT	5000	Used to change the port that a merchant uses when sending requests to ST Xpay	port=5001

2.5.2 Port security

ST Xpay uses a number of TCP/IP ports to communicate with SecureTrading and the merchant's system. The ports used by ST Xpay are detailed in the table below:

Number	Description
5000[1]	This port is used by the merchant to connect to ST Xpay when sending requests. This port must be closed to all external access. It is recommended that only localhost connections are made on port 5000.
80	This port is used to download ST Xpay core components when ST Xpay is started on the merchant's system. Any data downloaded will be authenticated via the java security manager before being run on the merchant system.
6666	This port is used to send encrypted requests and receive encrypted responses from SecureTrading. It must be available for both incoming and outgoing requests.

2.6 ST Xpay security considerations

When performing Xpay requests a certificate will be used to establish a secure connection to one of the SecureTrading payment servers.

---

1 Note that the internal port number 5000 is the Xpay default. If the merchant uses an alternative port number via the configuration file then the port security considerations must apply to that port instead of the default, i.e. ensure that the chosen port is not available externally.

It is important that adequate security measures are adhered to in order to safeguard ST Xpay certificates and prevent certificates from becoming compromised and used by others for illicit use.

When handling ST Xpay certificates it is recommended to follow these guidelines:

- Physically protect the certificates from unauthorised access
- Restrict access to the certificates to the fewest persons necessary
- Ensure that expired certificates are deleted and not kept on company systems

If you suspect that a certificate may have been compromised, contact support@securetrading.com immediately.

2.7 System requirements

The latest version of ST Xpay requires JDK 1.4.2 to operate.

Please note that other versions of Java may run ST Xpay but are unsupported.

For more information on Java, please see the Java website: http://www.javasoft.com

The recommended hardware requirements are a 600 MHz processor with 256 MB of RAM. Typical platforms that run the API include IBM compatible computers running a Linux/Windows operating system.

Care must be taken to ensure that the ST Xpay API and associated files, including any data requested or returned by it, remain secure on the merchant's server.

It is recommended that credit card details are not stored on the merchant's system.

3 An example transaction

3.1 Example test.xml

To send any transaction using Xpay to SecureTrading for authorisation, you must create a string, adhering to the Xpay `XML` specification. Please refer to the SecureTrading XML Specification document for more information. To obtain a copy of this document please refer to the Additional information section in this document.

An example of an XML string for an Auth transaction can be found in `test.xml` located in the example directory.

Open `test.xml` in a text editor and set the `<SiteReference>` tag to be your Site Reference.

> E.g.   `<SiteReference>company1234</SiteReference>`

where `company1234` is your business's Site Reference. You may also need to update the `<ExpiryDate>` tag to a date in the future.

Note: *check* that your Site Reference is entered exactly as quoted in your email. Your Site Reference is *case sensitive*. See Prerequisites if you don't know your Site Reference.

3.2 Communicating with Xpay

Most programming languages can be used to communicate with the Xpay client, such as Perl, Python, Java, C, C++, Cold Fusion, ASP etc.

SecureTrading has supplied an example of how to use the Xpay client. This example can be found in the `example` directory, the source code for which is `Test.java`

`Test.java` reads the XML file, `test.xml` and sends it to the Xpay client. It then displays the value returned by the Xpay client. The binary version of `Test.java` can be found in the example directory as `Test.class`

To run this example, open another command prompt and change to the `example` directory. Your current directory (`.`) must also be in the CLASSPATH. See the Troubleshooting guide (section 5) for help.

Now run the example `Test.java`
The first parameter should be the path to your certificate file, as sent to you by SecureTrading.

> E.g.   `c:`
> `cd \xpay\example`
> `java Test company1234certs.pem`

Note: the above command is *case sensitive*.

Running `Test.java` automatically extracts the details of your certificate file (`company1234certs.pem` for example) and adds it into the example `XML` string, `test.xml`
The `XML` strings that your script will create must also add in this information. See section 4 for details.

3.3 Result

The XML output should be:

```
Reading from file: company1234certs.pem
Reading from file: test.xml
Please wait while the transaction is authorised...
Waiting for response
```

A pause of approximately 2 – 20 seconds follows, (depending on your internet connection speed), before a string containing well-formed XML is returned as the output:

```
<ResponseBlock Live="FALSE" Version="3.51">
  <Response Type="AUTH">
    <OperationResponse>
      <TransactionReference>15-9-1124245</TransactionReference>
      <TransactionCompletedTimestamp>2009-04-14 14:25:12</TransactionCompletedTi
mestamp>
      <AuthCode>AUTH CODE:TEST</AuthCode>
      <TransactionVerifier>AQ+BBlSEdY1rx1DSFrXdVUwGnyZDd2TBltw+gyEGzaTOJgtGk/ND4
53t4EEPf0qDGe/kbPH6cVyKCvoWxb1QlOMHjggc9lf0fTUKhPv3z6WdLazYmIlNdox8cWXtYnAVwZ3ce
RfQl1s4nC7NwRlCQOUFf32UzYSrV1s+C2MVDsBg=</TransactionVerifier>
      <Result>1</Result>
      <SettleStatus>0</SettleStatus>
      <SecurityResponseSecurityCode>2</SecurityResponseSecurityCode>
      <SecurityResponsePostCode>2</SecurityResponsePostCode>
      <SecurityResponseAddress>2</SecurityResponseAddress>
    </OperationResponse>
    <Order>
      <OrderInformation>This is a test order</OrderInformation>
      <OrderReference>Order0001</OrderReference>
    </Order>
  </Response>
</ResponseBlock>
```

The XML returned contains the document root (<ResponseBlock> </ResponseBlock> tags). Within the XML string there will be the tag <Result>, the value of which will be numeric. If this number is "1" then the transaction was successfully authorised (except this is just a test and no real credit card details will be authorised), and you have now demonstrated a test transaction through the SecureTrading Payment Gateways using the Xpay client.

If the result is not "1", then the test transaction was not successfully authorised.
If you obtained a different result or encountered any problems then please refer to the Troubleshooting guide. See section 5.

For further information on the Xpay XML protocol, please refer to ST XML Specification.

In the example, Test.java displays the returned information to the screen. Test.java is just an example of what to do with the returned information. In reality, it is more likely that you will write your code to store and process the resulting string, rather than simply display it to screen. See ST XML Specification for more details.

4 Your personalised program

The diagram below describes how and where Xpay fits with the payment process.
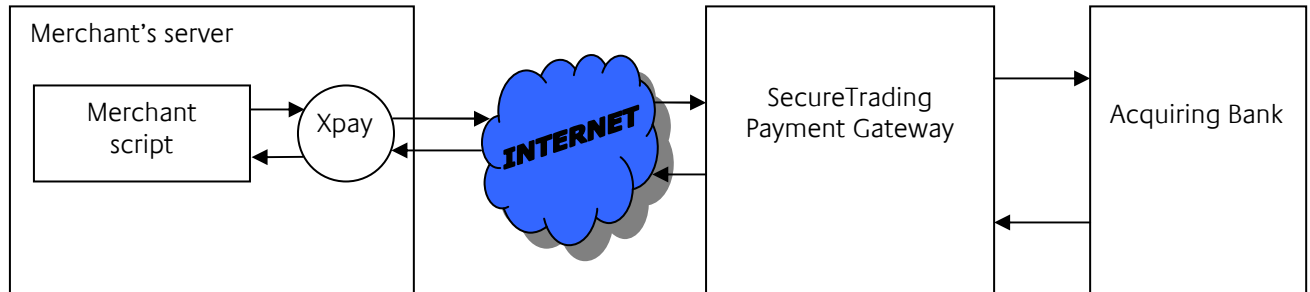

Figure 1: Xpay in the payment process

The SecureTrading ST Xpay API currently facilitates the secure processing of real-time credit card transactions and allows the management thereof.

The diagram above represents the flow of events when performing requests through ST Xpay.

To initiate a communication link to SecureTrading the following steps are involved.

- The ST Xpay client is hosted on the merchant's secure system
- The merchant's system creates the required XML string to be used by ST Xpay
- The merchant sends XML String to ST Xpay via internal TCP/IP socket[2]
- The merchant waits for response from ST Xpay on the open socket. ST Xpay returns a plain text string containing XML
- The merchant closes socket
- The returned data is processed by the merchant

Important

Only a single ST Xpay XML request must be sent through any particular socket.

4.1 Start the client

As in the Test example, you must have the Xpay client running.

4.2 The XML string

To explain how best to write your application it is best to follow similar steps to the program in the example see An example transaction. The source of the example can be found in `Test.java`

First create an `XML` string. Again, as before this can be done in any suitable programming language. See ST XPay API for the specification of valid XML requests.

As mentioned in Communicating with Xpay, your `XML` string must contain your certificate, such that the `<Certificate>` tag contains something similar to the following:

---

2 The default port to connect to is 5000. The port number is configurable by the merchant. For further details on port configuration, refer to C.

E.g.
```
<Certificate>
-----BEGIN CERTIFICATE-----
XXXXXXXXXXXXXXXXXXXXXXXXXXX . . .
. . . XXXXXXXXXXXXXXXXXXXXXXXXXX
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXX . . .
. . . XXXXXXXXXXXXXXXXXXXXXXXXXX
-----END RSA PRIVATE KEY-----
</Certificate>
```

Note: for testing purposes it is OK to "copy and paste" your certificate from your certificate file (`company1234certs.pem`, for example) into your `XML` string. However, for any live system you should find another way of storing the certificate file (such as in a secure database). Then when your system generates the `XML` to send the certificate can be extracted and added to the `XML`.

When sending a request to SecureTrading it may be required that customer details' including their personal information is required. Credit card numbers are used in the `XML` string so *must* be obtained in a *secure* manner, such as by the *https* protocol.

SecureTrading does *not* recommend storing credit card details on your servers. Refunds and Repeat transactions can be carried out using the `TransactionVerifier` (see ST Xpay API document) thereby removing the need to store card numbers.

4.3 Internal socket

Within your application, you will need to open an internal socket to pass your `XML` string to the Xpay client. This can be done in two ways.

Note: both methods involve the `XML` string being passed internally to the specific computer so the information will *not* be available or posted onto the internet. Therefore at this point no encryption is required.

4.3.1 Using a standard TCP/IP socket

This is recommended for programs written in, for example, Java, Perl or Python. The `XML` string is passed to the `localhost` with the address `127.0.0.1` This socket should use `port 5000`.

E.g. the following 'program' is pseudo-code to show the normal steps in instantiating a socket to the Xpay client:

```
\\ Import the socket software
import Socket

\\ Define the xml string (see The XML string)
xml = "<ResponseBlock><..></..></ResponseBlock>"

\\ Instantiate a new socket with IP Address of 127.0.0.1 and Port 5000
s = new Socket("127.0.0.1",5000)

\\ Send the xml down the socket
s.send(xml)
```

```
\\ Receive the response from XPay
result = s.receive()

\\ Close the socket
s.close()

\\ Result is now available for processing by the merchant
```

The data stored in `result` is a string containing XML, i.e. the information returned from the Payment Gateway.

4.3.2 Using the HTTP Post method
This is recommended for programs written in, for example, Cold Fusion. The `XML` string is POSTed via an internal URL.

The equivalent URL that would be entered into an internet browser would be similar to:

`http://127.0.0.1:5000/?xml=<ResponseBlock><..></..></ResponseBlock>`

I.e. Host Address of `127.0.0.1` and Port of `5000`. To create the `XML` string to be POSTed, it is unlikely that you will want to enter it by hand into an internet browser. A more suitable way could be to program the equivalent in, for example, Cold Fusion:

```
<cfhttp method="post" url="http://127.0.0.1/" port="5000">
<cfhttpparam encoded="No" TYPE="cgi" NAME="xpay" VALUE="#xml#">
</cfhttp>
```

This assumes that a variable named "`XML`" has already been created with the relevant `XML` string (see The XML string).

4.4 Xpay to SecureTrading Gateways and back

Once the `XML` string has been passed internally to the Xpay client, there is no need for anything else until the Xpay client receives the result from a Payment Gateway. The response echoes the authorisation status (authorised or declined) as decided by the acquiring bank. The Xpay client handles encryption and authentication itself.

See the ST XML Specification document for a more detailed outline of the operations between the Xpay client, the Payment Gateways and the acquiring bank.

4.5 Xpay result

This result passed to the Xpay client is then returned down the same internal socket. Once this string is available, you can parse the results. In the example `Test.java`, the result is simply displayed on-screen. However, you can use the result in any way you want. For example, upon a successful result being parsed, you could display a specific `.html` page confirming the success of the transaction to your customer, and/or send an appropriate email etc.

4.6 Other operations

Any other operations, such as refunds, settlements etc are approached in a similar way, by creating an `XML` string and sending it internally to the Xpay client to be encrypted and sent to a payment Gateways. As before the result of the operation will be returned as a string containing `XML`. See the ST XML Specification document for more information on other operations.

5 Troubleshooting

If you have any problems with your application, please try following the example *before* contacting SecureTrading. By following the trouble-shooter and reviewing your problems, you may be able to solve them yourself quicker and easier than by contacting SecureTrading.

If Xpay does not run, check:
- Xpay has an authenticated auto-update feature. Please ensure your server has access to the internet. You may need to open port 80 in your firewall if you have one and you should ensure you are running Xpay under a java security manager.
- Please ensure you are using the version of Java as specified in the Prerequisites
- Java must be installed correctly, possibly with the correct PATH set.
- We recommend you have e*xactly one* Xpay client running on your server.

If the `Test` example fails, check:
- Your Site Reference must be entered correctly in `Test.java`
- The Xpay client must be running concurrently while run the Test.
- You must specify your certificate file as a first parameter when running the Test.
- You must have a valid Certificate obtained from SecureTrading.
- Check that the current directory (`.`) is set in the CLASSPATH. It should be by default, but you may need to add this in. e.g. "`set CLASSPATH=.`"
- If the test fails to connect to a gateway, you may need to open port 6666 on your firewall if you have one. Alternatively if you have more than one version of java installed on your server your Xpay client may be using the wrong one. To overcome this when starting the Xpay client specify the complete java file path e.g. `"C:\Program Files\Java\j2re1.4.2_14\bin\java" –Djava.security.manager – Djava.security.policy=c:\xpay\xpay\examplepolicy XPay`

If your own application fails, check:
Please first try the `Test` example. If you can get the `Test` example to work successfully, as described in An example transaction, then the problem is probably with your application's code, so consult your own application user manual.

5.1 Possible error messages

`java.net.BindException: Address already in use`
This means another services is running on the same port as Xpay. It could be due to another Xpay client already running. Alternatively, you can run Xpay on another port. By default, port 5000 is used by Xpay. To use a different port, see the configuration file section in ST Xpay API Specifiaction document.

`java.security.AccessControlException: access denied`
This means the configuration of the policy file and/or the keystore is incorrect. Ensure that your text editor has not appended an extension such as ".txt" to the files, and that all the paths to each file in startup script and the policy file are correct.

5.2 XPay characters

Xpay currently supports only printable us-ascii characters (i.e. those in the range 32 to 127). Any characters outside this range should either be replaced with an alternative character (many encodings use "?" for this) or encoded into bytes which are representable in ascii using your own choice of encoder. (Under certain conditions, characters with a byte value of greater than 128 may process correctly, but this should not be relied upon)

In addition, the XML specification states that certain characters must be encoded before being entered between XML tags. Most XML parsers will do this step automatically for you, but the specific required encodings are:

"<" should be replaced by "<" (without the quotes)

"&" should be replaced by "&" (without the quotes)

Also, it is recommended (although not required) that single (') and double (") quotes are replaced with ' and " respectively and that end-tag characters (>) should be replaced by >

6 Additional information

6.1 Support

SecureTrading provides support for its software and the operation of its payment service.
If you require technical support, first ensure that you have read and understood all relevant documentation.
Please also attempt the `Test` example as outlined in An Example transaction

If the problem persists, please email support@securetrading.com, quoting your SecureTrading sitereference and concisely stating the nature of your problem.

To help us help you, please include the original XML string sent and any error messages that are returned by the ST Xpay API verbatim.

Note: Before sending any information to SecureTrading, care should be taken to remove any sensitive information, such as the credit card number.

SecureTrading additional contact details:

Phone: 01248 672 050
Fax: 01248 672 099

6.2 Further reading

For further information please refer to the following documents:

In the general setup guides (http://www.securetrading.com/support/general-setup-guides.html) section of the SecureTrading website:
- Going live document:
- SecureTrading testing document:

In the ST Xpay documents (http://www.securetrading.com/xpay.html) section of the SecureTrading website:
- SecureTrading Xpay user guide
- SecureTrading XML Specification

Bundled with the ST Xpay distribution:
- ST Xpay read me: readme.txt

New features and request types will be added to ST Xpay. Information on these features will be included in newer versions of this document or separate documentation will be provided. The Xpay section can be found on the SecureTrading web site: http://www.securetrading.com/xpay.html

7 Glossary of terms

Authorisation

Authorisation is the process of validating a credit card transaction with an acquiring bank. Authorisation allocates the transaction amount on a customer's credit card but no money is debited from the customer's credit card account.

Good password

A good password is a password containing characters from more that two different character types, i.e. lower-case letters (d), upper-case letters (D), numbers (2), special characters (@).

Payment Gateway

A payment gateway can be thought of as a secure, reliable bridge from a merchant's web site or server to the acquiring banks.

Refund

A refund is a two-stage process. Firstly, a request is given to transfer monies from a merchant's account to the customer's credit card account for a previously settled transaction. The money is merely allocated; no money is credited to the account until the second stage. Secondly, the monies are transferred from the merchant to the customer's credit card account.

Settlement

Settlement is the process of debiting the transaction amount from a customer's card and into the merchants account. Typically settlement follows an authorisation or refund request.

String

A sequence of characters, e.g. 'Hello World.' The maximum permitted string length when passing an XML string to SecureTrading is 255 characters.

XML

eXtensible Markup Language provides a structured method of defining data.
For further information on XML, please refer to http://www.xml.org/